Amendments to the Specification

Please replace the paragraph **[0004]** with the following rewritten paragraph:

[0004] When updating one index key column for the database table, the down level application must delete the affected row from the table so that the down level application may enter the modified data into the column at-issue. Upon updating the index key column, the down level application would provide the record to an index manager within the database engine to reinsert the updated record in the index according to the index ordering method, e.g., a B-trec index. In the prior art, the down level application would delete and update a record by first fetching the record, deleting the record, updating the columns and then reinserting the record. However, if the down level application does not provide data for all the columns, then the database engine would use default values for extended columns not recognized by the down level application in the inserted record, thereby erasing any meaningful data that was previously maintained in the extended columns by the upgraded version but unknown to the down level application.

Please replace the paragraph **[0019]** with the following rewritten paragraph:

[0019] FIG. 3 illustrates an example of a record data structure 70 the application server 6 communicates back to the application client 4b to provide information on a requested record in the ~~table 22~~ table 14. Upon accessing a requested record in response to a query, the application server 6 would generate the record data structure 70 to include a unique record identifier (ID) 72 of the accessed record in the table 14 and data from the columns in the accessed record into corresponding column fields 74a, 74b....74n.

Please replace the paragraph **[0020]** with the following rewritten paragraph:

[0020] FIG. 4 illustrates logic implemented in the application clients 4a, 4b....4n and application server 6 (FIG. 1) to process a request from an application client 4a, 4b....4n to update a key column that is used to order the records in the index 16. In certain implementations, the application client 4a, 4b....4n will update a key column of an index 16 for one table 14 by first fetching the record to update, deleting the record from the database table 14, updating the record, and then reinserting the record in the table 14 to cause an index record to be generated and inserted into the index 16 according to the key column(s) in the generated index record and the ordering defined for the key column(s) of the index 16. In one implementation, control begins at block 100 in the application client 4a, 4b....4n to update data in the table by performing a fetch/delete/reinsert combination of steps. The application client 4a, 4b....4n queries (at block 102) the table 14 for records and selects all of the columns of the requested records. If the application client 4a, 4b....4n was the initial version 50 and the table 14 was the subsequent version 62 including all seven columns (C1-C7), then the initial version application client 50 would only select the first five columns (C1-C5) because the initial version of the ~~application 50~~ application client 50 does not view extended columns (C6, C7) in the subsequent version 62 of the table 62. However, if the querying application client 4a, 4b....4n is the subsequent version 60, then the application 4a, 4b...4n would select all of the columns (C1-C7) in the subsequent version 62 of the table 62.

Please replace the paragraph **[0021]** with the following rewritten paragraph:

[0021] At block 110, the application server 6 receives a request from the application client 2a, b..n for data in the selected columns. In response, the application server 6 executes (at block 112) the query to access the requested record. A record data structure 70 (FIG. 3) is then generated (at block 114) for each of the requested columns including data in the column fields

74a, 74b....74n and the record ID 72. The application server 6 determines (at block 116) whether the application client 4a, 4b....4n requested all the columns from the table. This determination may be made by accessing information on the client version 50 or 60 provided when the session with the client was initiated. Client version information may indicate whether the particular client version recognizes less table columns than the current application version of the application server 6. Additionally, when establishing the session, the client application 4a, 4b....4n may communicate the table 14 columns the application 4a, 4b....4n recognizes. The application server 6 can then determine whether the number of client recognized table columns is less than the actual number of columns in the ~~table 22~~ table 14. If (at block 116), the requesting client application 4a, 4b....4n recognizes the same number of columns as currently implemented in the table 14, then the application server 6 returns (at block 118) the generated record data structure 70 (FIG. 3) to the requesting client application 4a, 4b....4n.

Please replace the paragraph **[0026]** with the following rewritten paragraph:

[0026] In such implementations, the application client 4a, 4b....4n interfaces with the application server to backup data objects at the client system 2a, 2b...2n to the application server 6, which in turn may maintain data in various archive and backup storage devices. The application server 6 maintains ~~database tables~~ databases 12 providing information about data objects archived by the application server 6, inventory resources, such as available storage resources, network resources, and information on the configuration of network resources accessible to the application server 6. Those skilled in the art will recognize that the described implementations for updating records in a database through a fetch/delete/insert combination can apply to any client/server application.